KNOWLEDGE INTERNALIZATION IN PAIR PROGRAMMING PRACTICES

School of Computing
UUM College of Arts & Sciences
Universiti Utara Malaysia

mawarny@uum.edu.my[1]
mazni@uum.edu.my[2]
mazida@uum.edu.my,[3]

Khairul Bariah Ahmad

School of Multimedia Technology & Communication
UUM College of Arts & Sciences
Universiti Utara Malaysia

kbariah@uum.edu.my

ABSTRACT

Pair programming practice has been widely used as a pedagogical approach in educational setting specifically in the programming course. Most pair programming studies agree that this practice can foster knowledge sharing among students. However, the studies do not highlight knowledge internationalization during pair programming practice. Therefore, this paper will discuss knowledge internalization based on tacit knowledge that occurs from knowledge sharing activities in pair programming practices. This is achieved by employing the process of Socialization, Externalization, Combination, and Internalization (SECI) in the form of learning, thinking and decision-making skills among the students. 119 participants were actively engaged in the pair programming practice in this study. The participants were required to answer questionnaires, which were adapted from the SECI model to suit the educational context. Statistical t-test was used to analyse the data. The results showed that pair programming was able to promote knowledge internationalization in the thinking process. This study contributes to a better understanding of important knowledge sharing activities to construct student's skills

during the internalization process through pair programming. Future works will be focused into a rigorous theoretical framework for constructing tacit knowledge among the students in pair programming environment.

Keywords: Pair programming, knowledge sharing, internalization, tacit knowledge.

INTRODUCTION

Learning a programming course is generally considered hard, difficult and often contributes to high dropout rates among students. "Too bad, too hard, easy to understand the concept but difficult to write the programme" are some common reasons that have been given by students about the programming course. Many innovative approaches had been introduced to overcome this problem. Innovation in pedagogical approaches and programming tools used to assist teaching and learning of programming courses were introduced in order to provide a positive impact on students' performance (Abdullah, 2006). Thus, pair programming is one alternative used as a pedagogical tool in teaching and learning programming courses.

Pair programming as one of the key practices in Extreme Programming has been gaining acceptance among practitioners and the software development community. This success leads to wide use of pair programming in the educational setting as a computer science or software engineering pedagogical tool especially in programming courses (Canfora, Cimitile & Visaggio 2003; Brereton, Turner & Kaur 2009, Cliburn, 2003; Mendes, Al-Fakhri & Luxton-Reilly 1997). Various studies have been done on determining the usefulness and effectiveness of pair programming as a pedagogical tool and the following positive results were indicated:

1. Pair programming can improve students' performance by gaining higher scores on programming assignments (Werner, Hanks & McDowell 2004; McDowell, Werner, Bullock & Fernald, 2003; Cliburn, 2003; Slaten, Droujkova, Beenson, Willioms & Layman, 2005).
2. Pair programming can increase student's confidence and satisfaction (Werner et al., 2004; McDowell et al., 2003; Cliburn, 2003; Slaten et al., 2005).
3. Pair programming can encourage students to complete the programming course (Werner et al., 2004; McDowell et al., 2003).

Pair programming shifts programming learning from a solitary activity into a collaborative learning process (McDowell et al., 2003). It involves two students who act as a driver and navigator working on the same problem from design to the testing phase. In general, the driver is the person who is involved in creating and implementing the code, whereas the navigator is responsible for checking the errors and suggesting the implementation technique. The navigator provides an alternative solution to the given problem and assists the driver to solve the problem. Meanwhile, the driver fully controls all input through the keyboard or mouse and comes out with solutions based on his/her idea or the navigator's suggestions (Williams & Kesler, 2000; Beck, 2005).

Besides the roles, switching partners is an important issue that should be considered in implementing pair programming. Switching partners and role rotation can induce knowledge sharing among students (Chau & Maurer, 2004; Beck, 2005). This leads to an exchange or spreads information and knowledge throughout the whole team of software development (Muller & Tichy, 2001). Indeed, a better structured pair interaction is required by having proper communication within a pair (Gallis, Arisholm & Dyba 2003; Beck, 2005). Pair programming involves an informal and spontaneous communication as it relies on face-to-face communication between the driver and the navigator (Chau & Maurer, 2004). However, frequent switching of partners is required in achieving knowledge sharing (Gallis et al., 2003).

Pair programming can foster knowledge sharing among students. Pair programming is usually performed by students as novice programmers to develop small programming tasks, which improve knowledge transfer and quality (Vanhanen & Korpi, 2007). Many studies have been done with pair programming in education, however, most of them do not highlight internalized knowledge particularly tacit knowledge from the knowledge-sharing processes between students who act as drivers and navigators in the pair-programming practice. Thus, this study discusses knowledge internalization based on the knowledge sharing activities in pair-programming practices by employing the process of Socialization, Externalization, Combination and Internalization (SECI). For completeness, the overview of knowledge sharing will be discussed in more detail in the next section. Then, this study will cover the method used in the implementation of this study. The last section describes the result and discussion of this study.

KNOWLEDGE SHARING

Knowledge management is an important consideration in software development to ensure that knowledge flows efficiently among software development team

members. Knowledge management is more than the centralized repository project data and information (Komchaliaw & Wongthongtham, 2010). Knowledge management is needed to properly manage the knowledge shared within the software-development team involving all software stakeholders. In a broad sense, knowledge management is defined as a multidisciplinary paradigm which uses technology to enable the creation, codification, transfer and application of knowledge in the organization (Nonaka, 1991; Gover & Davenport, 2001). Knowledge transfer is an important part of knowledge management and promotes knowledge transmission among individuals in a community or an organization. Generally, knowledge transfer involves knowledge sharing from the starting point of knowledge creation to the point of knowledge application (Gover & Davenport, 2001). According to Hsia et al. (2006), knowledge transfer is also obtained by adapting several cyclic activities namely mobilizing knowledge, knowledge searching, knowledge distributing, knowledge sharing, and knowledge pulling and pushing.

Generally, knowledge management covers the obtaining process, sharing, utilizing and storing knowledge among individuals in an organization. Knowledge sharing is an important part of knowledge-management and a crucial task in the agile software development processes. It promotes the knowledge-transmission among individuals in the community or the organization and normally is supported by the knowledge-sharing mode (Fengjie, Fei & Xin, 2004). There are various kinds of knowledge management modes that enable individuals to exchange knowledge such as face-to-face communication, conference, knowledge network, and organization learning. However, this study focuses on the face-to-face communication as a knowledge sharing mode in co-located pair programming practices.

According to Fengjie et al. (2004), the knowledge-sharing process involves two main parties namely the contributor and the receiver. Figure 1 depicts an overview of the knowledge-sharing process which involves the two parties and the process of knowledge transmission. The contributor contributes a part of his/her knowledge and transmits it to the receiver. The receiver will receive the knowledge and try to add his/her understanding and transform it into his/her knowledge. This scenario is similar to the pair-programming practices where the navigator plays the role of a contributor and the driver the role of a receiver. Then navigator will provide suggestions in assisting the driver to solve the given problem in implementing the programme whereas the driver will use the suggestions given and blend them with his/her own knowledge to come out with the best solution. Knowledge sharing in pair-programming practices involves communication, updates, advice, problem-solving, decision-making, discussion over project data and information (Komchaliaw & Wongthongtham, 2010).

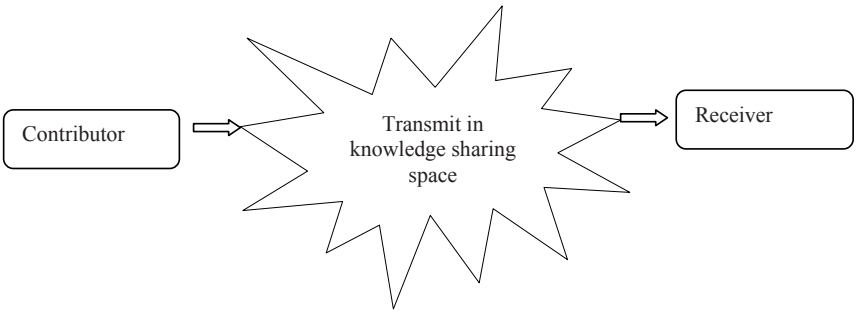

Figure 1. Knowledge sharing process.

Three steps are involved in transmitting fluent knowledge. Firstly, the receiver is not just the knowledge beneficiary but also the knowledge provider during the knowledge-sharing process. In pair programming, the tendency to avail knowledge-sharing should be overcome by promoting it as a reward (Yin & Zhang, 2005). In a learning environment, a grade satisfaction is a reward to encourage them to share their knowledge to achieve a good solution in programming. Besides, a well-designed knowledge-sharing space is required to assist knowledge transition such as NetMeeting, Yahoo/MSN Messenger, web-based knowledge-sharing system and others. There is no restriction of time and place especially to implement the distributed pair programming. Students need the virtual collaboration environment when their schedules conflict and they cannot get physically together in finishing the programming assignment (Ho et al., 2003). Lastly, a proper way is needed to ensure the knowledge is easy to understand by having an effective communication.

During the pair-programming process, some explicit and mostly tacit knowledge is shared between the driver and the navigator (Chau & Maurer, 2004). Explicit knowledge is easy to share because it can be expressed in words and numbers (Nonaka & Konno, 1988; Ho, Raha, Gehringer & Williams 2003; Fengjie et al., 2004). However, the representation of explicit knowledge which is easy to understand and convenient to retrieve should be considered during the explicit knowledge-sharing process (Fengjie et al., 2004). Meanwhile, more efforts are required to gain tacit knowledge because it is very hard to formalize and difficult to codify tacit knowledge. Tacit knowledge is human judgment and strategic decision making (Brockmann & Simmonds, 1997; Guthrie, 1995). The main sources of tacit knowledge are experience and thinking (Gerard, 2003). Tacit knowledge is related to the teaching and learning process and is also generated through the learning experience (Gerholm, 1990). Thus, tacit knowledge will be obtained through pair-programming practices between pairs to generate learning, thinking and decision-making skills.

Opposed to explicit knowledge, tacit knowledge is hard to share due to its difficulty to express it in language (Fengjie et al., 2004). Thus, Socialization, Externalization, Combination and Internalization (SECI) are adopted in this study to facilitate knowledge conversion between tacit and explicit knowledge and also to promote knowledge-sharing between partners during pair-programming practice. Socialization is a process of sharing experiences and thereby creating tacit knowledge such as shared mental models and technical skills. Externalization means the process of articulating tacit knowledge into written form or explicit knowledge but still in inconsistent condition so that it can be shared by others and become the basis of new knowledge. Combination refers to the process of converting explicit knowledge that is inconsistent into a more complex and systematic set of explicit knowledge. During the internalization process, systematic explicit knowledge will be converted into tacit knowledge. The experience acquired through the previous process is converted into valuable knowledge for individuals and organizations (Nonaka & Takeuchi, 1995). Normally, knowledge internalization is referred to as "knowledge application capability" which focuses on the ability to apply the particular knowledge in a real situation. However, students should have "knowledge creation capability" which leads them to create new knowledge to solve programming tasks during knowledge internalization (Nonaka & Takeuchi, 1995; McElroy, 2000).

In pair-programming practices, knowledge-sharing involves social interaction, sharing and constructing knowledge between the partners. The SECI model is applicable to promote sharing and constructing tacit knowledge between partners in generating learning, thinking and decision-making skills. Thus, this paper discusses on internalization based on the knowledge-sharing activities in pair-programming practices by employing the process of SECI. The factors investigated were types of internalized tacit knowledge in the form of learning, thinking and decision-making skills among the students. In order to assess empirically the effect of knowledge-sharing amongst programmers using pair programming, the following hypotheses has been formulated:

- Ho: There is no difference in the state of learning activities in the internationalization of knowledge-sharing between pair programmers and non-pair programmers.
- H_1: There is significant difference in the state of learning activities in the internationalization of knowledge-sharing between pair programmers and non-pair programmers.
- Ho: There is no difference in the state of thinking activities in the internationalization of knowledge-sharing between pair programmers and non-pair programmers.

- H_1: There is significant difference in the state of thinking activities in the internationalization of knowledge-sharing between pair programmers and non-pair programmers.
- Ho: There is no difference in the state of decision-making activities in the internationalization of knowledge-sharing between pair programmers and non-pair programmers.
- H_1: There is significant difference in the state of decision-making activities in the internationalization of knowledge-sharing between pair programmers and non-pair programmers.

METHOD

Procedure and Sample

The sample of the study consisted of undergraduate College of Arts and Sciences (CAS) students at Universiti Utara Malaysia (UUM) enrolled in the *Basic Programming* course. The *Basic Programming* course is a compulsory course for first year students in information technology (IT), multimedia, and education in IT. Each week students attend two hours of lectures and a two-hour laboratory session. In the laboratory, students are required to solve programming assignments assigned by the lecturer. The students were divided into two groups; pair programming group and non-pair programming group to work on the assignments. During the lab session, an instructor was assigned to assist and support the students to solve programming problems for both groups.

In the midst of the semester, 119 questionnaires were distributed to the students who were actively engaged in pair-programming practices and had experience applying non-pair activities. Students were required to complete a ten-minute survey to determine the level of knowledge-sharing amongst pair and non-pair programmers. All questionnaires were returned completed, representing an acceptable response rate. Of the 119 questionnaires administered, 77 students from the pair-programming groups and 42 from the non-pair programming groups completed the survey. To ensure the validity of the knowledge-sharing scores, outliner data was excluded in the analysis, resulting in a data set of 118 respondents. The age of the respondents ranged from 20 to 25 years, with a mean age of 18.7 years. Slightly more than 65.5% of the respondents were females.

Measure

In order to test the hypotheses, a survey study was conducted. The questionnaire was adapted from the SECI model in a educational context (Mazida, 2010) particularly focusing on the internalization factor. The validity and reliability

of this questionnaire was demonstrated in an other study (Mazida, 2010). The factors investigated were the types of internalized tacit knowledge in the form of learning, thinking and decision making skills among the students. All items in the questionnaire were measured using a five-point Likert scale ranging from "1-Strongly disagree", "2-Disagree", "3-Don't know", "4-Agree", and "5-Strongly agree".

An independent t-test was conducted to measure the level of knowledge-sharing between the pair-programming and non-pair programming groups. The independent t-test was used to compare the two groups' level of knowledge-sharing in terms of learning, thinking and decision-making skills between these groups. The SPSS tool was used to analyse the data. Reliability analysis for this questionnaire was 0.7, which exceeds the minimum requirement of Cronbach Alpha, 0.6 (Nunally, 1978).

RESULT AND DISCUSSION

Data was analysed in terms of learning, thinking and decision-making skills. The main goal was to demonstrate that pair-programming practice is a viable tool to promote knowledge-sharing activities, particularly amongst the programming students.

Learning

There was no significant difference in the score of learning for pair programming ($\underline{M}$ =20.24, $\underline{SD}$=3.40), and non-pair programming groups [$\underline{M}$ =19.43, $\underline{SD}$=3.76; $\underline{t}(116)$=1.19 , $\underline{p}$=0.24]. This is illustrated in Table 1.

Table 1

Group Differences for Learning between Pair Programming and Non-pair Programming Groups

	Pair Programming		Non-Pair Programming		
Learning	$\underline{M}$	$\underline{SD}$	$\underline{M}$	$\underline{SD}$	$\underline{T}$
	20.24	3.40	19.43	3.76	1.19

*$\underline{p} < 0.05$

In educational context, pair programming involves two novices that need guidance from the lecturer as an expert. Since the data was from new first-year students, who were exposed to the programming course, it was not

surprising that they still needed guidance from their lecturers in solving the programming tasks. They needed time to understand and gain knowledge of the programming concepts and applications. Even though they could do the tasks, facilitation from the lecturer exceeded what could be attained in pair work (Vygotsky, 1978). The level of the students' potential ability was uplifted to a higher level with guidance from the experts compared to self-learning (Holtzman, 2009). This finding is supported by Heywood et al. (1992) and Berg, Bergendahl and Lundberg (2003) who claimed that students require strong support from the instructor for a better education. With continuous guidance from the lecturer, the lecturer's tacit knowledge is transferred and the knowledge is shared amongst the pair. This finding suggests that expert knowledge from the lecturers is important in guiding students to acquire the required knowledge in the fundamental programming course. Different and imbalanced levels of programming skills and knowledge (Katira et al., 2004; William et al., 2006; Hahn et al., 2009) between the partners also have an impact on the learning process. The partner with higher knowledge in programming may prefer to work alone as he/she feels that working alone can quicken his/her programming tasks, whereas the partner with a lower level of knowledge requires support from his/her partner to learn and solve the tasks. This has shown that the lecturer needs to take into account the student's knowledge level before selecting the partners. By having a balanced knowledge level between the pairs, the learning process in pair programming can be facilitated, and thus internationalization process can be achieved.

Thinking

There was significant difference in the score of thinking for pair programming ($\underline{M}$ =20.24, $\underline{SD}$=3.40), and non-pair programming groups [$\underline{M}$ =19.43, $\underline{SD}$=3.76; $\underline{t}(116)$=2.47 , $\underline{p}$=0.015]. This is illustrated in Table 2.

Table 2

Group Differences for Thinking between Pair Programming and Non-pair Programming Groups

	Pair Programming		Non-Pair Programming		
Thinking	$\underline{M}$	$\underline{SD}$	$\underline{M}$	$\underline{SD}$	$\underline{T}$
	18.97	2.26	17.90	2.25	2.47*

*$\underline{p} < 0.05$

Pair programming activities require support from a partner in solving programming problems. Therefore, this activity will be stimulating both programmers to think and exchange ideas during the activity. This in

line with the findings of Williams et al. (2002) who claimed that students applying pair-programming showed higher order thinking skills compared to solo programmers. This is because they could share knowledge to solve programming assignments with their partner during pair programming. Therefore, the students are more independent in their thinking without fully relying on the instructor to discuss solutions during lab sessions. In addition, by doing programming in pairs, tacit knowledge instilled in the brain can be transferred among the students, which encourages intrinsic motivation among team members (Mazni et al., 2009). In pair programming, students need to be alert and attentive to check and review their partner's code programme. This situation encourages them to think more compared to non-pair programmers, which develops codes in isolation. When this happened, logical thinking amongst the pair increased and assisted them to broaden their way of thinking, which improved the internationalization process during programming activities.

Decision-making

There was no significant difference in the score of decision-making for pair programming ($\underline{M}$ =17.82, $\underline{SD}$=2.28), and non-pair programming groups [$\underline{M}$ =17.76, $\underline{SD}$=1.97; $\underline{t}(116)$=0.13 , $\underline{p}$=0.9]. This is illustrated in Table 3.

Table 3

Group Differences for Decision-making between Pair Programming and Non-pair Programming Groups

Decision Making	Pair Programming		Non-Pair Programming		
	$\underline{M}$	$\underline{SD}$	$\underline{M}$	$\underline{SD}$	$\underline{T}$
	17.82	2.28	17.76	1.97	0.13

*$\underline{p} < 0.05$

In this study, the internationalization process of decision-making refers to students' independence of making decisions in their learning process. Chong et al. (2005) noted that pair programming promotes better decision-making when two heads are better than one. However, in reality, this position is not always true. Students in pair have to put more effort in creating mutual understanding between them in order to make better decisions. In addition, they need to lean and consider appropriate action taken by their partner to solve the programming tasks. Although pair programming promotes the exchange of ideas and knowledge between pairs, internationalization process

cannot be achieved due to the reliance of the pair to make decisions. Unlike solo programmers, students are more independent in making decisions based on their individual knowledge level. In order for pair programming to achieve higher internationalization in the decision-making process, investigation into pair characteristics such as programming ability and personality types (Hannay et al., 2010; Hahn et al., 2009; Katira et al., 2004; Williams et al., 2006) can be carried out. Common characteristics of pair partners are important to drive consensus in the decision-making. Nevertheless, programmers need to understand each partner's differences to reach project goals successfully. Understanding others' differences yield more added values and better decision-making process in programming tasks.

From the results, it can be seen that both pair programming and non-pair programming achieved a statistically significant result in thinking activities. Therefore the null hypothesis H_0 for thinking activities in the internationalization of knowledge-sharing has been rejected. However, two null hypotheses, H_0 for learning and decision-making in internationalization of knowledge-sharing has been accepted because there were no statistically significant difference for both groups.

CONCLUSION

This study contributed to better understanding of important knowledge-sharing activities to construct students' skills during the internalization process through pair programming. It is undisputed that pair programming is one of the pedagogical approaches that can enhance students abilities in the areas of programming. Pair programming is able to promote internationalization in the thinking process because both programmers are actively involved in solving programming tasks. However, students as novice programmers still need guidance from the experts, who are their lecturers to improve their learning process. In terms of decision-making skills, students that are involved in pair-programming activity are relying on their partners on making decisions. Therefore, they need to achieve consensus before finalizing their programming tasks. To ensure that internationalization in the decision-making process can be increased in pair programming, investigating the level of programming knowledge and personality types of the students can be carried out. Knowledge-sharing in pair programming can be improved with the guidance of lecturers and also by increasing the frequency of programming activities between the pairs. Socialization factors a such as meeting daily is an important factor in ensuring the success of pair programming. Pair programmers need time to understand the other's differences. This can lead

them to share insights, lead their thoughts, make sound decisions, and thus induce knowledge-sharing during programming activities. Further work to be considered can be a rigorous theoretical framework for constructing tacit knowledge among the students in pair programming environments.

REFERENCES

Abdullah, M. Z. (2006). Improving learning of programming through e-Learning by using asynchronous virtual pair programming. *Journal of Distance Education*, *7*, 162-173.

Beck, K. (2005). *Extreme programming explained: Embrace change* (2nd ed.). Reading, Mass: Addison-Wesley.

Brereton, P., Turner, M., & Kaur, R. (2009). Pair programming as a teaching tool: A student review of empirical studies. *Proceedings of the Conference on Software Engineering Education and Training*, 22, 240-247.

Brockmann, E. N., & Simmonds, P. G. (1997). Strategic decision making: The influence of CEO experience and use of tacit knowledge. *Journal of Managerial Issues, IX*(4), 454-467.

Canfora, G., Cimitile, A., & Visaggio. C. A. (2003). Lessons learned about distributed pair programming: What are the knowledge needs to address? *Proceedings of the Twelfh IEEE International Workshops on Enabling Technologies: Infrastructure for Colloborative Enterprises.*

Chau, T., & Maurer, F. (2004). Knowledge sharing in agile software teams. *Lecture Notes in Computer Science, 3075*/2004, 173-183.

Chong,J. et al. (2005). Pair programming: When and why it works. *17th Workshop of the Psychology of Programming Interest Group*. Brighton, UK.

Cliburn, D.C. (2003). Experiences with pair programming at a small college. *Consortium for Computing Science in College, 19*(1), 20-29.

Fengjie, A., Fei,Q., & Xin, C. (2004). Knowledge sharing and web-based knowledge-sharing platform. *Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business.*

Gallis, H., Arisholm, E., & Dyba, T. (2003). An initial framework for research on pair programming. *Proceedings of the International Symposium on Empirical Software Engineering.*

Gerard, J. G. (2003). Measuring knowledge source tacitness and explicitness: A comparison of paired items. *Proceedings 5th Annual Organizational Learning and Knowledge Conference*.

Gerholm, T. (1990). On tacit knowledge in academia. *European Journal of Education*, *25*(3), 263-271.

Gover, V., & Davenport, T. H. (2001). General perspectives on knowledge management: Fostering a research agenda. *Journal of Management Information Systems, 18*(1), 5-21.

Guthrie, S. (1995). The role of tacit knowledge in judgement and decision-making. *Proceedings of the International Conference on Outdoor Recreation and Education*, 105-115.

Hahn, J. H. et al. (2009). Assessment strategies for pair programming. *Journal of Information Technology, 8*, 273-284.

Heywood, J. (1992). The training of student-teachers in discovery methods of instruction and learning and comparing guided discovery and expository method: Teaching the water cycle in geography. Technical Report, *Research in Teacher Education Monograph,* 1/92. Dept. of Teacher Education, Dublin University.

Ho, C., Raha, S., Gehringer, E., & Williams, L. (2003). *Sangam – A distributed pair programming plug-in for eclips*. Retrieved 1 August 2010 from http://collaboration.csc.ncsu.edu/laurie/Papers/Sangam.pdf.

Ho, C. W. (2003). *Tacit knowledge management and pair programming. CSC591m Term Paper 2003.* Retrieved 1 July 2010 from http://www4. ncsu. edu/~cho/articles/TCMandPP.pdf

Holzman, L. (2009). *Vygotsky at work and play*. NY: Routledge.

Katira, N. et al. (2004). On understanding compatibility of student pair programmers. *ACM Technical Symposium on Computer Science Education (SIGCSE)* Norfolk, VA, 7-11.

Komchaliaw, S., & Wongthongtham, P. (2010). A state of the art review on software project performance management. *4th IEEE International Conference on Digital Ecosystem and Technologies (IEEE DEST 2010),* 653-655.

Mazida, A. (2010). *An investigation of knowledge creation processes in LMS-supported expository and PBL teaching methods* (Unpublished doctoral dissertation). Universiti Sains Malaysia.

Mazni, O. et al. (2009). Being agile in classroom: An improvement to learning programming. *Seminar Kebangsaan ICT dalam Pendidikan*. Ipoh, Malaysia.

McDowell, C., Hanks, B., & Werner, L. (2003). Experimenting with pair programming in the classroom. *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, *35*(3), 60-64.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. *Proceedings of the 25th International Conference on Software Engineering*, 602-607.

McElroy, M. W. (2000). Integrating complexity theory, knowledge management and organizational learning. *Journal of Knowledge Management*, *4*(3), 195-203.

Mendes, E., Al-Fakhri, L. B., & Luxton-Reilly, A. (1997). Investigating pair-programming in a 2nd year software development and design computer science course. *Proceedings of the ITiCSE'05*, 285-295.

Muller, M., & Tichy, W. (2001). Case study: Extreme programming in a university environment. *Proceedings of the 23rd International Conference on Software Engineering,* 537-544.

Natarajan, G., & Shekhar, S. (2001). *Knowledge management: Enabling business growth*. Singapore: McGraw-Hill International Edition.

Nonaka, I. (1991). *The knowledge creating company*. Harvard Business Review, 69(6), 96-104.

Nonaka, I., & Konno, N. (1988). The concept of ba: Building a foundation of knowledge creation. *California Management Review*, *40*(3), 40-55.

Nonaka, I. & Takeuchi, H. (1995). *The knowledge creating company: How Japanese companies create the dynamics of innovation*. Oxford: Oxford University Press.

Nunnally, J. C. (1978). *Psychometric theory*. NY: McGraw-Hill.

Slaten, K. M., Droujkova, M., Beenson, S. B., Williams, L., & Layman, L. (2005). Undergraduate student perceptions of pair programming and agile software methodologies: Verifying a model of social interaction. *Proceedings of the Agile Development Conference. Software Engineering,* 36, 61-80.

Sommerville, J., & Craig, N. (2006). *Implementing IT in construction,* NY: Taylor and Francis Group.

Vanhanen, J., & Korpi, H. (2007). Experiences of using pair programming in an agile project. *Proceedings of the 40th Hawaii International Conference on System Sciences.*

Vygotsky, L. S. (1978). *Mind in society.* MA: Harvard University Press.

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *ACM Journal of Educational Resources in Computing, 4*, 1(3).

Williams, L. et al. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education,* 12, 197-212.

Williams, L. et al. (2006). Examining the compatibility of student pair programmers. *Agile Conference 2006*. Minneapolis, MN, 411-420.

Williams, L. A., & Kessler, R. R. (2000). The effects of "pair-pressure" and "pair-learning" on software engineering education. *Thirteenth Conference on Software Engineering Education and Training*, 59-65.

Yin, T. S., & Zhang, Q. (2005). Dynamic game analysis in worker's tacit knowledge sharing process in enterprise. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics,* Guangzhou.